Quality Aware Query Scheduling in Wireless Sensor Networks

Hejun Wu Qiong Luo Jianjun Li Alexandros Labrinidis

Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong, China
$\{$*whjnn,luo*$\}$*@cse.ust.hk*

School of Computer Science
Huazhong University of Science and Technology, Wuhan China
jianjunli@smail.hust.edu.cn

Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260-916
labrinid@cs.pitt.edu

Abstract

We study query scheduling in Wireless Sensor Networks (WSNs) with a focus on two important metrics: Quality of Service (QoS) and Quality of Data (QoD). The motivation comes from our observation that most WSN scheduling techniques ignore the quality requirements of queries. As a result, they are inefficient or inapplicable to quite a few applications that have different quality requirements. In this paper, we propose a distributed Quality Aware Scheduling (QAS) *framework to address this problem. QAS works on top of existing* quality-unaware *query scheduling protocols and allows individual users to specify their QoS and QoD requirements on their queries. Given these quality requirements, QAS determines the target qualities to be provided in scheduling and the execution order of these queries so as to maximize the total system profit. Our preliminary results show that QAS significantly outperforms the baseline scheduling algorithms in terms of system profit.*

1. Introduction

In-network query processing techniques for wireless sensor networks have been widely accepted for on-line sensory data management in WSNs [14], [25]. Instead of pulling all the sensory data from nodes into a central server to process, these techniques make the sensor nodes to cooperatively process the SQL queries from users within the network, hence can increase the flexibility and reduce the network traffic of WSNs [14]. Scheduling schemes have also been proposed to guarantee the efficiency and reliability of in-network sensor query processing [4], [9], [12], [21], [24]. These scheduling schemes are able to coordinate the timings of nodes to avoid communication collisions, to sleep nodes to save energy, and to make receivers active when a transmitter transmits packets to them, since a node cannot receive data when it is sleeping.

However, as far as we know, existing scheduling or query processing techniques ignore the quality requirements from different applications and users. For instance, in a WSN, a user may require 80% nodes to report their results within some predefined length of period, while another user may need query results from all of the nodes in order that the scientific data would not be misleading. Current scheduling schemes or query processing systems process such kind of queries in an ad-hoc way: queries with different quality preferences are equally treated and are processed one by one according to their arrival order. As a result, those queries with high quality requirements are likely to be unsatisfied, whereas the lower request queries that arrive earlier than the high request queries may be overly satisfied.

To address the above problem of these quality-unaware scheduling protocols, we propose a distributed Quality Aware Scheduling framework (QAS) in this paper. In QAS, users can specify their quality requirements on queries by giving revenues to different qualities in quality functions [8], [17]. Given the quality function, a WSN attains a profit from each processed query in accordance with the quality it served. The profit is the ratio between the attained revenue and the query processing cost. With the quality functions of the queries, QAS tries to find the best qualities and processing order of the queries to get the maximum profit for the underlying scheduling protocol. This profit is only the highest one for the current underlying protocol, but may not be the highest for others due to the efficiency differences among various scheduling protocols.

QAS is designed to run on top of a scheduling protocol and to apply the quality-aware scheduling strategy to this protocol. Given a certain level quality of a query on a node, QAS uses a cost model to calculate the needed cost and the revenue will be gained. In the cost model, the energy cost and response time each is a function of a set of network parameters. These parameters indicate the current status of the nodes such as number of hops, children, queries, etc. They are extracted from historical statistics of query processing and communication of the network. With this cost model, QAS can mathematically compute the quality level that can get the maximum profit from processing each query. When there are multiple queries running simultaneously, QAS will rearrange the order of these queries towards the maximum total system profit, since the resources of a WSN are too limited to satisfy the requirements of all queries.

As far as we know, QAS is the first distributed scheduling approach to address the problem of maximizing the system profit while at the same time, satisfying different quality requirement of users in WSNs. Moreover, it contributes to the distributed query scheduling study in sensor networks with the definition of network parameters and cost model. Its work flow enables the workload and served quality to be evenly distributed on each node of a WSN. Finally, QAS allows the quality result feedback for dynamic schedule adjustment to further improve the system profits in a network. We performed a series of experiments to compare QAS with other scheduling methods. Experiment results show that, powered by the three quality-aware scheduling mechanisms, QAS significantly outperforms the baseline algorithms in terms of system profit and capacity.

The remainder of this paper is organized as follows. Section 2 describes the background of query processing and scheduling, defines the scheduling problem for maximizing the quality profits in query processing, and then reviews the related work on QoS and QoD. Section 3 describes the overview of QAS. Section 5 details the design and implementation of QAS. The evaluation results of QAS compared to other base line schemes are shown in Section 6. Finally, we conclude this paper and list some future work directions in Section 7.

2. Background and Related Work

In this section, we first present the network, data, query and scheduler in a query processing system, we then define QoS, QoD, and system profit in such a system, and finally we review the related work. The symbols (excluding those commonly used, e.g., U - voltage, the temporary variables, and those in the algorithm) used throughout this paper is summarized in Table 1.

Table 1. Summary of symbols used

Symbol	Definition
D	Number of query results received by the sink
A	Number of all query results generated in a WSN
$D(i)$	Number of transmitted query results in hop i
$A(i)$	Number of all results should be generated in hop i
D_Q	Data quality on a node
H	Maximum number of hops
I_h	Electric current in hibernating (sleeping)
I_p	Electric current in computation
I_r	Electric current in receiving
I_t	Electric current in transmitting
L	Query lifetime
N_c	Average number of children
N_d	Average number of descendants
$N(i)$	Number of nodes in hop i
r	Ratio of parallel transmission among all nodes
T_a	Aggregating time on a node
T_c	Communication time on a node
T_h	Hibernating time
$T(j)$	Execution time of the j-th operator in a query
T_{Np}	Network processing time of a query
T_{Nc}	Network communication time of a query
T_{Nq}	Network query evaluation time of a query
T_o	Overlap between the operations on a node
T_q	Query evaluation time on a node
T_r	Time for receiving data from the children of a node
t_s	Length of a time slot
k_1, k_2, k_3	Coefficients of quality functions
b_1, b_2, b_3	Coefficients of quality functions

2.1. Preliminaries and Problem Formulation

2.1.1. Network. A Wireless Sensor Network (WSN) is an ad hoc, multi-hop, wireless network that is composed of spatially distributed nodes, each of which is equipped with a CPU, radio and sensing devices to cooperatively monitors the physical characteristics of a target [19]. Such a WSN has three features as follows: (1) the nodes are highly limited in hardware resources [14], [27]; (2) the nodes use multi-hop communication to report their data due to the short transmission range and limited transmission power of each node [6]; and (3) the communication may be overwhelmed by collisions if nodes transmit packets simultaneously.

Existing query processing systems use the tree network structure due to its simplicity, duplicate-freeness and communication efficiency [14], [25]. In addition, the systems assume the networks on which they are running to be stationary. In a tree network, the sink node is the *root*. All the other nodes should report their query results, if any, to this root node. The *parent* of a node serves as the router that forwards or aggregates the data of the node. A *neighbor* of a node is one within this node's transmission/receiving range. By this definition, the parent of a node is also its neighbor. Finally, a *sibling* means the node that shares the same parent with one node.

A query processing system on a WSN is shown in Fig. 1. In this network, the server and the sink node together constitute the base station. A user posts one or more queries on the server, which injects these queries into the network via the sink node [14], [25]. The other sensor nodes then start to process and schedule these queries, generate query results, and report their results back to the sink. In a query processing system, the scheduler arranges both the starting time and the execution order of operations [24].

2.1.2. Data. Each sensor node in a WSN maintains an attribute table that specifies types of sensory and non-sensory data [14]. The sensory data, such as temperature, light, etc., can be acquired from sensing devices. The non-sensory data, such as node ID, hop count, children and neighbor table, etc., is the attribute of a node and doesn't need to be acquired by sensing devices. Table 2 shows these two kinds of data and their properties on a TelosB mote.

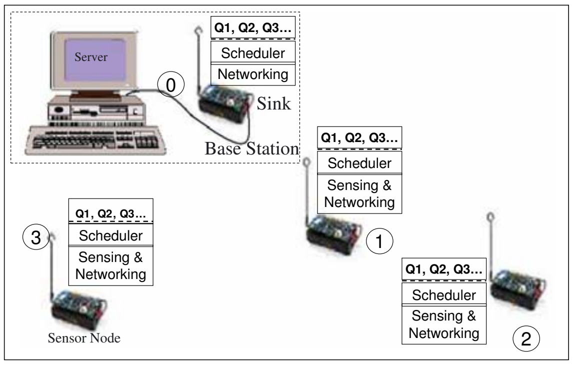


Figure 1. Query processing in a WSN

Table 2. Sensor data

Data	Range
Temperature	-40 ° C to 123.8 °C
Humidity	0 to 100% RH
Light	320 nm to 730 nm
Non-sensory data	Depending on the network and node

2.1.3. Query. Currently, in-network query processing systems mainly support two kinds of read-only queries: acquisition query and aggregation query [13], [14], [27]. An acquisition query uses projection and selection to collect the needed sensory data while an aggregation query uses database aggregates such as AVERAGE, MAX, SUM, COUNT, to get the collective information of the entire network or a group of nodes [14]. According to whether the queries reside in each node continuously sample the sensory data or not, the queries can also be divided to continuous query and snapshot query. A continuous query requires that the nodes sample the sensory data once every fixed interval, which is called a sample interval or epoch [14]. A snapshot query only needs to sample the sensory data once.

$Q1$ gives an example of a continuous acquisition query. This query requires the nodes in a WSN to report their light readings if the light reading is less than λ, which can be a threshold value defined in applications. $Q2$ shows an example of an aggregation query, which requires the nodes to report the average temperature every 60 seconds.

Q1:
SELECT nodeid, light
FROM sensors
WHERE light $< \lambda$
Q2:
SELECT AVG (temperature) FROM sensors
WHERE temperature $< \tau$
SAMPLE INTERVAL 60s

A query result refers to the result that is received at the sink node. It is usually composed of data collected from the sensor nodes or is an aggregated value about the sensed data from nodes in a WSN. Note that according to this definition, data packets transferred from a sensor node to the sink is not a query result, it is only a part of the result when it arrives at the sink.

2.2. Scheduling for Query Processing in WSNs

There have been great efforts from researchers in scheduling the communication of the WSNs. FPS [9], SS [21] and DCS [24] are the representative scheduling protocols. These protocols can either be directly used or be adopted for query processing. There are also some other schemes for event detection [4] or data collection that needs only one time wake-up per epoch [12]. These protocols provide the slot allocation to reduce wireless competition and the idle listening periods. The problem with them is that they are not quality-aware and thus are not applicable for queries with quality requirement.

The major assumption in these scheduling protocols is time synchronization. Currently, there are many realistic and efficient time synchronization protocols [7], [22]. In addition, scheduling protocols require all nodes injected queries report their results within the allocated transmission slots. In case the predicate of a query can not be satisfied on a node, the node should report a short message to its *parent* to indicate that it does not have the result for this query. This enables a node to know whether its children have finished query result reporting and consequently each node is able to know the quality of its children.

2.3. Problem Definition

2.3.1. QoS. In this paper, QoS refers to response time and/or query lifetime in query processing. The response time is the period from the start time of query processing to the time when all of the nodes have reported their query results in one epoch (sample interval). The query lifetime is described using the number of epochs from the time of query injection to the time when the query stops running. The lifetime requirement of a snapshot query is 1, since a snapshot query needs only one epoch of processing. The reason of including these two performance metrics in QoS is as follows: some queries desire short response time, e.g., event monitoring queries, some queries prefer long query lifetime, e.g., data collection queries, and some queries may require both short response time and long life time, e.g., queries in factory or health monitoring applications.

2.3.2. QoD. In QAS, QoD is defined by Equation (1). In this equation, D is the number of query results received by the sink of a WSN, while A is the number of all query results generated in the WSN. If A is 0, i.e., there is no satisfying query result in the network, then $\frac{D}{A} = 1$.

Note that $\frac{D}{A}$ alone does not describe the data quality of a network well: A WSN may want to return query results from nodes that are closer to the sink node as much as possible to save energy. In this scenario, the value of $\frac{D}{A}$ will still be high but the data quality under such scenario may be low in effect, since there are few results from the nodes that are far from the sink. To avoid such a problem, Equation (1) uses the average weighted difference between $\frac{D}{A}$ and $\frac{D(i)}{A(i)}$ of each hop i", $\rho(\frac{D}{A} - \frac{D(i)}{A(i)})$, as a punishment. Here $D(i)$ is the number of transmitted query results in hop i, H is the maximum number of hops, and $A(i)$ is the number of all results that should be generated in hop i. Similarly, if $A(i)$ is 0, which indicates that there is no satisfying query result generated in hop i, then $\frac{D(i)}{A(i)} = 1$.

$$QoD:\ D_Q = \frac{D}{A} - \frac{1}{H}\zeta\sum_{i=1}^{H}(\frac{D}{A} - \frac{D(i)}{A(i)}) \qquad (1)$$

In Equation (1), ζ is a user specified weight ($0 < \zeta \leq 1$ and $\zeta = 0$ when $\frac{D}{A} \leq \frac{D(i)}{A(i)}$). A larger ζ indicates that a user expects the received source data to be more evenly distributed among hops.

2.3.3. Quality Award. Similar to traditional databases, users may accept a range of QoS and QoD, although they desire them to be as high as possible. Since providing higher QoS and QoD in query processing correspondingly needs more cost (energy and time), users should specify the awards to the attained qualities. The QoS and QoD awards are usually described using "money" [17]. The amount of money is determined by a user defined quality-award function.

Fig. 2 shows a set of example of quality-award functions. In this figure, the X axis is the provided quality and the Y axis represents the award given to the corresponding quality. The total awards from a query processing is $AwardtoQoS + AwardtoQoD$.

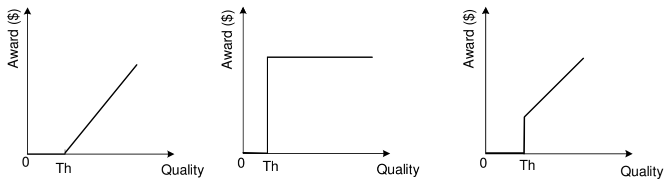


Figure 2. QoS/QoD award functions

2.3.4. System Profit. As mentioned, the system profit is the ratio of award and the cost. Due to the severe limitations of hardware and energy resources in WSNs, the energy cost is critical for a sensor network to attain the highest award during its lifetime. Hence, we define the system profit gained from a query as Equation 2, where φ is the price of battery energy, with a unit of dollar per joule ($\$/joule$). The problem for QAS is to maximize the profit of the queries processed.

$$Profit = \frac{AwardtoQoS + AwardtoQoD}{\varphi \cdot cost} \qquad (2)$$

2.3.5. Scheduling Problem. With the definition of system profit, we now define the scheduling problem for QAS. Given a network $G = (V, E)$, s is the sink node, $s \in V$. If two nodes i, j, $i \in V$, $j \in V$, can communicate, then i, j can be denoted as (i, j), $(i, j) \in E$. Suppose that a number of queries $Q = \{q_1, q_2, \ldots, q_n\}$ have been installed on each node, the problem for QAS is to maximize the total system profit from processing a subset of queries in Q, as stated below,

$$max : S = \sum_{k=1}^{m} Profit(q_k)\ (q_k \in Q, m \leq n) \qquad (3)$$

Although QoS and QoD are well studied in traditional database areas [1], [2], [5], [8], [15], [17], there is no existing work that studies QoS and QoD in query scheduling for wireless sensor networks yet. Due to the different conditions and requirements, current studies on scheduling in traditional databases are not directly applicable to query scheduling in WNSs. However, these studies of QoS and QoD in traditional databases still provide helpful reference for the problem of profit maximization in WSNs. Additionally, previous scheduling protocols [4], [9], [21], [24] for both computation and communication are tightly related to our work here. In the following, we review some representative scheduling studies in both traditional databases and sensor networks.

2.4. Related Work

Previous work on scheduling in WSNs [4], [9], [21], [24] for both computation and communication are tightly related to ours, as QAS is designed to work on top of them. FPS [9], SS [21] and DCS [24] are the representative scheduling protocols. These protocols can either be directly used or be adopted for query processing. There are also some other schemes for event detection [4] or data collection that needs only one time wake-up per epoch [12]. These protocols allocate slots so as to reduce wireless competition and the idle listening periods. The problem with them is that they are not quality-aware and thus are not applicable for queries with quality requirement.

Although QoS and QoD scheduling are well studied in traditional database [1], [2], [5], [8], [15], [17], there are few studies on QoS and QoD in query scheduling for wireless sensor networks yet. However, these studies of QoS and QoD in traditional databases share a common goal of profit maximization with our work.

Qu *et al.* proposed the concept of quality contract to integrate QoS and QoD metrics [17]. With quality contract, the scheduling scheme is able to perform the tradeoff between QoD and QoS to maximize the total system profit. We adopted the concept of quality contract in designing our QAS. In Borealis [1], Abadi *et al.* proposed a QoS model that aggregates multiple metrics with different weights to be a single metric for evaluating the QoS. We adopt this concept of total system profits in QAS.

There are extensive studies on priority (or value) based and deadline oriented scheduling in the database community. For instance, Haritsa *et al.* proposed Value over Relative Deadline (VRD) to enable the queries whose deadlines are closer to the current time to be executed earlier [8]. The purpose is to complete as many queries as possible. These deadline and value based scheduling methods are mainly used in real-time databases [10], [15], but they are also helpful in query scheduling of WSNs where queries post fixed deadlines for result reporting.

Recently, there are initial investigations in query processing quality of WSNs. For instance, Amirijoo et al. defined the sensory data quality as the length of the sample interval in continuous data collection [3]. The authors proposed mechanisms to lengthen the WSN lifetime by dynamically adjusting the sampling period. However, their definition is not applicable to continuous queries with sample interval specified already [14]. Yates et al. defined the data quality as the normalized delay and proposed an approximation approach to reduce the delay [26]. These essentially focused on the query processing delay (QoS) as defined in this paper.

There are also two representative studies [18], [16], on the quality of data (QoD) similar to the QoD defined in this paper. Among these, Ren et al. proposed an algorithm to select the most related nodes as the active nodes to answer queries in a WSN to reduce the energy consumption of nodes without undermining the data quality much. Peng et al. elaborated the assessment models of data quality in in-network data processing. Their methods are quite helpful to improve the admission control and the active node selection in QAS, although they did not consider query and communication scheduling. We are planning to adopt and extend these methods as our future work.

To the best of our knowledge, QAS is the first work that considers both QoS and QoD in scheduling of WSNs. Next, we will show how QAS improves the QoS and QoD while observing energy efficiency.

3. Scheduling Framework Overview

Before coming to the details of the cost model in QAS, we present the overview of QAS to show its general idea. Fig. 3 illustrates the architecture of QAS.

On each node, the protocol array adopts a set of existing scheduling protocols to schedule the query operators and communications of each query [9], [21], [4], [24]. For a scheduling protocol to be loaded to the array, we design a

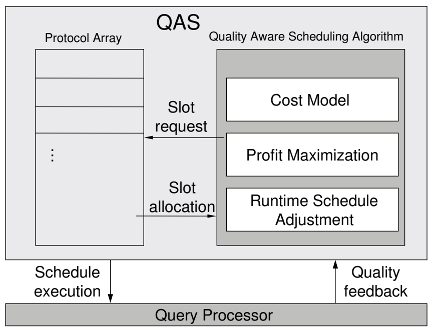


Figure 3. QAS architecture

uniform interface that allows QAS to specify the number of transmission, receiving and query execution slots for each query. This way, QAS is able to control the target quality of each query to be scheduled by the protocols. The underlying protocol to run is system specified before query processing.

In case a protocol does not have operator scheduling support sometimes, QAS will construct a sensor query operator tree for the protocol. In such a query operator tree, there are mainly three types of query operators: sampling, selection, and aggregation. Note that QAS regards the operation of fetching non-sensory data as sampling operator too, since the purpose of this operation is also to get attributes. Projection operator is excluded, since only the queried attributes will be fetched in QAS. Fig. 4 shows a query evaluation tree based on this definition of sensor query operator. It can be seen that the query evaluation tree gives the order and types of operators to be executed in processing the query.

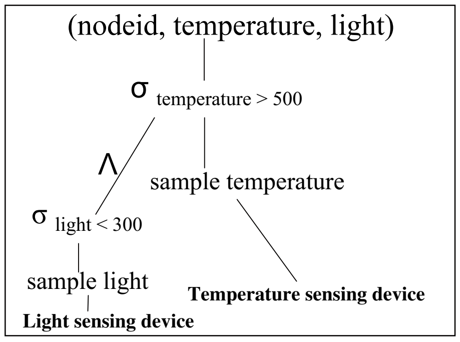


Figure 4. QAS on an example WSN

After the query plan is constructed, the quality-aware scheduler determines the quality and execution order of queries for the underlying protocol using a cost model so that the WSN attains the total maximum profit. To avoid the scenario in which the target qualities of nodes differ much due to this distributed quality determination, it uses the same set of network parameters (see Section 4) in the cost model on nodes. Extending our model to allow nodes to

Table 3. Execution Time of Sampling Operators

Type	Delay(ms)
Temperature	226
Humidity	78
Light	22
Non-sensory data	0.1

have different parameter settings is a challenging direction of future work. The details of the cost model will be presented in the next section.

4. Cost Model

The cost model in QAS estimates the network processing time and the energy consumption of a single query. The motivation of this model is to express the relation between the QoS, QoD, time and energy costs, and the system profit of a query, so that QAS is able to determine the target QoS and QoD to get the maximum system profit.

4.1. Network Processing Time

The network processing time, T_{Np}, is divided into query evaluation time, T_{Nq}, and communication time, T_{Nc}. The network query evaluation time of a WSN, T_{Nq}, is called network query evaluation time, which is the accumulated time of non-overlapping evaluation time on the nodes of the WSN. Similarly, T_{Nc} of a WSN is the accumulated time of non-overlapped communications on all nodes of the WSN. By definition, the time when a node transmits or receives messages in the midst of query evaluation is considered in T_{Nc}, not T_{Nq}.

T_{Nq}: We first investigate the query evaluation time on a single node, T_q so that we can calculate T_{Nq} with T_q. T_q is calculated as $T_q = D_Q \sum_{j=1}^{n} T(j) - T_o$, where D_Q is the QoD to be provided by this node, $T(j)$ is the execution time of the j-th operator and T_o is the overlapping time between the query execution and the communication on the node.

The operator execution time can be measured off line. We have measured three sampling operators for sensory data: temperature, humidity, and light sensing on Telos B motes [6]. We also measured the sampling operator for fetching the non-sensory data. The execution time for these operators are listed in Table 3.

In a WSN, the nodes in upper hops are usually started at the same time or later than lower hop nodes [14], [4]. Therefore, the query evaluation time of the upper hop nodes and the query evaluation of the lower hop nodes will overlap, or the query evaluation time and the communication of other nodes will overlap. The non-overlapping network query evaluation time thus is roughly the same as that of a single node, i.e., $T_{Nq} \approx T_q$.

T_{Nc}: We estimate T_{Nc} by Equation (4). In (4), D (Directly forward) refers to a node that directly forwards the results from its children towards the sink; A (Aggregate) refers to a node that aggregates the results of its children first and then reports the aggregated result. The equations in the remainder of this paper use the same denotations. In the two formulas for $forward$ and $aggregate$, r is the ratio of parallel transmission among nodes in the network. r is used to remove time of simultaneous transmissions on the nodes. $N(i)$ is the number of nodes in hop i and H is the maximum number of hops in the network. t_s is the length of a slot. D_Q is the QoD. The reason of using D_Q in the $forward$ formula is that D_Q determines the number of packets to be forwarded on each node. N_c is the average number of children and T_a is the average time for aggregating a result. r, $N(i)$, H, N_c, and T_a are called the *network parameters*, which are managed by the sink and disseminated to the other nodes before query processing.

$$T_{Nc} = \begin{cases} (1-r)D_Q t_s \sum_{i=1}^{H} iN(i), & D, \\ N_c T_a + \sum_{i=1}^{H} N(i), & A. \end{cases} \tag{4}$$

Lemma 1: Given a WSN whose routing paths are fixed, the network processing time of a query, T_{Np}, is a linear function of data quality D_Q: $T_{Np} = \alpha D_Q + \beta$.

Proof: As described above, $T_{Np} = T_q + T_{Nc}$. From the equations of T_q and T_{Nc}, T_{Np} can be estimated as follows.

$$T_{Np} = D_Q \sum_{j=1}^{n} T(j) - T_o + \begin{cases} (1-r)D_Q t_s \sum_{i=1}^{H} iN(i), & D, \\ N_c T_a + \sum_{i=1}^{H} N(i), & A. \end{cases} \tag{5}$$

In (5), once the network and the scheduler is fixed, the network parameters such as H, $N(i)$, t_s, and T_o are all constants. Denoting these constants using α and β as shown in Equations (6) and (7), we have $T_{Np} = \alpha D_Q + \beta$, hence the lemma follows.

$$\alpha = \sum_{j=1}^{n} T(j) + \begin{cases} (1-r)\sum_{i=1}^{H} iN(i)t_s, & D, \\ 0, & A. \end{cases} \tag{6}$$

$$\beta = -T_o + \begin{cases} 0, & D, \\ N_c T_a + \sum_{i=1}^{H} N(i), & A. \end{cases} \tag{7}$$

□

4.2. Energy Consumption

The average energy consumption for processing a query on a node in a WSN can be estimated from the node running time and electric current: $E = U \cdot I \cdot T$, where E is the energy consumption of a node within T length of time, during which the voltage and the electric current of the node are U and I, respectively. Considering the different operations in query processing, the energy consumption should be $E = U \sum I_j \cdot T(j) \cdot L$, where I_j and $T(j)$ are the electric current and the execution time of each type of the j-th operation per epoch, L is the total number of epochs in processing the query.

With the above analysis, the node energy consumption is modeled in Equation (8). In (8), T_q is the query evaluation time and T_o is the same as that in Section 4.1. I_p is the electric current of the computation for query evaluation. Usually, a node can turn off the radio chips to lower the electric current. T_r is the time for receiving the results from the children. I_r is the electric current of receiving on the node. T_c is the communication time as described in Equation (5). $T_c - T_r$ is the total transmitting time on the node. I_t is the electric current of transmitting. Finally, T_h and I_h are time and electric current in hibernating (sleeping), respectively.

$$E = (T_q I_p + (T_c - T_r) I_t + T_r I_r + T_h I_h) \cdot U \cdot L \qquad (8)$$

In Equation (5), the electric current of transmission in each epoch is assumed to be constant. This assumption is realistic because in the scheduling protocols and current query processing systems, the transmission power and the corresponding transmission range are all fixed in each epoch. In addition, the average retransmission time due to transmission failures is included in the transmitting time per epoch.

In Equation (5), T_h is calculated by removing the active time (T_q, T_c and T_r) from the total time. Suppose that the time length for transmitting and receiving a result is t_s, T_q, T_c and T_r are computed as follows. First, T_q: T_q is deduced in Section 4.1. Second, T_c: Suppose the size of communicated results is n, then T_c is $n \times t_s$. Hence, the task is to find n of a query on each node. Considering the characteristic in query processing, n is not constant in different queries. There are two scenarios for calculating this n: (1) If the query needs the node directly forward the results of its children, suppose the query result of each node is reported, then the node should receive N_d results, forward them and send an additional result to its parent. N_d is the size of the results from the descendants of this node. Considering the QoD to be provided, $n = D_q(2N_d + 1)$, where D_q is the QoD. (2) If the query requires the node to aggregate the results of its children, then $n = (N_c + 1)$, where N_c is the size of the results from the children. Here D_q is not included, since no matter what D_q is to be provided, each node has to aggregate a result from each of its children. Finally, T_r: Similarly to T_c, we can deduce that for an acquisition query, $T_r = D_q N_d t_s$, and for an aggregation query, $T_r = D_q N_c t_s$.

Given the above processing time on each node, the energy consumption can be expressed as $E = (\lambda D_Q + \delta)L$, . They are computed in Equation (9) and (10). N_d in Equation (9) is the average number of descendants of each node in the WSN. It is used to calculate the receiving and forwarding slots for the descendants.

$$\lambda = U I_p \sum_{j=1}^{n} T(j) + U \cdot \begin{cases} N_d t_s I_r + (N_d + 1) t_s I_t, & D, \\ 0, & A. \end{cases} \qquad (9)$$

$$\delta = -T_o U \cdot I_q + U \cdot \begin{cases} 0, & D, \\ N_c T_a I_p + N_c t_s I_r + t_s I_t, & A. \end{cases} \qquad (10)$$

5. Quality Aware Scheduling

We first show the details of how to maximize the system profit that a given scheduling protocol can attain, using the cost model presented in the last section. Then we present our quality aware scheduling algorithm that determines the quality to be realized for the given scheduling protocol.

5.1. System Profit Maximization

With the quality functions, the system profit P can be calculated by the following equation,

$$P = \frac{k_1 L + b_1 + k_2(T_E - T_{Res}) + b_2 + k_3 D_Q + b_3}{Cost} \qquad (11)$$

where $k_1, k_2, b_1, b_2, k_3, b_3$ are coefficients, L and T_{Res} is QoS, D_Q is QoD. $Cost$ is the average node energy consumption of the network. The response time T_{Res} can be computed by Equation (12), in which T_{last} is the current response time of existing queries. According to Lemma 1, $T_{network} = \alpha D_Q + \beta$. Hence, T_{Res} is also a linear function of D_Q, $T_{Res} = \alpha D_Q + \gamma$, where $\gamma = T_{last} + \beta$. We thus have,

$$T_{Res} = T_{last} + T_{network} = \alpha D_Q + \gamma \qquad (12)$$

Then, (11) can be easily converted to Equation (13). There are two variables, D_Q and L in (13). The profit maximization problem becomes searching for a pair of (D_Q, L) such that P is maximized. In essence, this is a problem of calculating extreme value of bivariate functions. We use the *partial derivatives* to solve this problem.

$$P = \frac{k_1 L + b_1 + k_2(\alpha D_Q + \gamma) + b_2 + k_3 D_Q + b_3}{(\lambda D_Q + \delta)L} \qquad (13)$$

Firstly, We can get the first partial derivatives of system profit in the direction of D_Q and L as shown in Equation (14) and (15). Usually $\frac{\partial P}{\partial L} \geq 0$ as $k_1 \geq 0$ and $\delta \geq 0$ and a node only needs to consider the changes of D_Q to get the maximum system profits from queries. It is obvious that $\frac{\partial P}{\partial D_Q} = 0$ and $\frac{\partial P}{\partial L} = 0$ is a necessary condition for (D_Q, L) to be the extreme value point of P.

$$\frac{\partial P}{\partial D_Q} = \frac{(k_2\alpha + k_3)\delta - (b_1 + b_2 + b_3 + k_2\gamma)\lambda - k_1 \lambda L}{(\lambda D_Q + \delta)^2 L} \qquad (14)$$

$$\frac{\partial P}{\partial L} = -\frac{b_1 + k_2(\alpha D_Q + \gamma) + b_2 + k_3 D_Q + b_3}{(\lambda D_Q + \delta) L^2} \qquad (15)$$

The second partial derivatives of system profit are calculated as follows:

$$A = \frac{\partial^2 P}{\partial D_Q^2} = 2\frac{(b_1 + b_2 + b_3 + k_2\gamma)\lambda^2 + k_1\lambda^2 L - (k_2\alpha + k_3)\delta\lambda}{(\lambda D_Q + \delta)^3 L} \quad (16)$$

$$B = \frac{\partial^2 P}{\partial D_Q \partial L} = \frac{(b_1 + b_2 + b_3 + k_2\gamma)\lambda - (k_2\alpha + k_3)\delta}{(\lambda D_Q + \delta)^2 L^2} \quad (17)$$

$$C = \frac{\partial^2 P}{\partial L^2} = 2 * \frac{b_1 + k_2(\alpha D_Q + \gamma) + b_2 + k_3 D_Q + b_3}{(\lambda D_Q + \delta)^2 L^3} \quad (18)$$

With the above partial derivatives, QAS is able to find out the point, (D_{q0}, L_0), that enables the system profit $Profit(q_i)$ of each query to be a maximum one by using the following mathematical method:

(1) Find the (D_{q0}, L_0) that makes $\frac{\partial P}{\partial D_Q} = 0$ and $\frac{\partial P}{\partial L} = 0$.

(2) If D_{q0}, L_0 satisfy the threshold quality of the query, then calculate the value of A, B and C at (D_{q0}, L_0), where A, B, C are computed by Equations (16) to (18); otherwise, goto (4).

(3) Use $M = AC - B^2$ to check whether (D_{q0}, L_0) is the maximum point, if $M < 0$, then (D_{q0}, L_0) is the maximum point. otherwise, we goto (4).

(4) If the maximum point is not found or does exceeds the quality threshold, D_{q0}, L_0 should be set as the lowest or the highest possible quality, depending on which one makes the larger system profit.

(5) Finally, QAS calculates the target response time, T_{R0}, from D_{q0} and get (D_{q0}, T_{R0}, L_0), which is the target quality for the query. QAS sorts the injected queries in the ascending order of $Profit(q_i)$. Then it schedules the query to attain the maximum profit from all queries.

5.2. Distributed Profit Maximization Algorithm

With the cost model and the partial derivatives in the previous sections, we now present our distributed algorithm for profit maximization in query processing. Algorithm 1 shows the steps of the determination of the target qualities and the execution order of queries and then the process of calling scheduling protocol in QAS. Note that Algorithm 1 is run on each node instead of on the sink to reduce the communication overhead, since the communication is more costly than computation.

In this algorithm, the first segment (Lines 1 - 6) uses the partial derivatives of the cost model to calculate the target QoS and QoD, $tQoS[i]$ and $tQoD[i]$ for the i-th query. The system profit would be $tP[i]$, if the target qualities $tQoS[i]$ and $tQoD[i]$ were realized, as shown in Line 3. Then Lines 4 -5 calculate the number of receiving slots needed, $sr[i]$, and transmission slots, $st[i]$, using the target quality $tQoS[i]$ and $tQoD[i]$. In these two lines, $Nr[i]$ and $Nt[i]$ are the number of needed receiving and transmission slots if $tQoD[i] = 1$, i.e., the number of receiving and transmission slots that the node would allocate previously without QAS.

Algorithm 1 Scheduling for Profit Maximization

```
Input  $n$ queries with quality functions;
Output  Query execution order of these queries and the target QoS and QoD for each query; schedule of each query;
1: for $i = 1$ to $n$ do
2:     find $tQoS[i], tQoD[i]$ for query $i$;
3:     compute the profit, $tP[i]$ of the i-th query under $tQoS[i], tQoD[i]$;
4:     $sr[i] = Nr[i] \cdot tQoD[i]$;
5:     $st[i] = Nt[i] \cdot tQoD[i]$;
6: end for
7: for $j = 1$ to NUM_STRATEGIES do
8:     sort the queries in the descending order of their weights, the order is denoted as $O[j]$;
9:     move the query in $O[j]$ whose quality requirement cannot be satisfied into the waiting list $w[j]$;
10:    $P[j]$ = the sum of system profit of queries in $O[j]$;
11: end for
12: find the execution order, $O[k]$, that has the highest profit ($1 \leq k \leq NUM\_STRATEGIES$);
13: send $O[k]$ and the $sr, st$ of the queries in $O[k]$ to the underlying scheduling protocol to build up the schedule for each query in $O[k]$;
```

The second segment (Lines 7 - 13) optimizes the query execution order of multiple queries to be scheduled and processed to get the maximum total system profit. This algorithm uses different weights to sort the queries and can get a number of orders of query execution of the set of queries as shown in Lines 5 - 9. QAS currently uses two strategies (NUM_STRATEGIES = 2) to define the weights: (1) maxP[i] and (2) maxP[i] / tQoS[i]. We found in our experiments that in most occasions, these two types of weights allowed QAS to get higher profits than a single type.

6. Evaluation

6.1. Experiment Setup

We ran the prototype of QAS on four scheduling protocols of WSNs, DCS [24], AHS [23], FPS [9], and SS [21]. We compared QAS with the following quality-aware scheduling schemes applied on the protocols: (1) Low-quality(low), which always serves the lowest acceptable QoD and life time of a query; (2) Medium-quality(Mid), which serves the medium level QoD and life time within the range of QoD and lifetime requests; (3) Random-quality(Rand), which randomly chooses the target quality from the range of QoD and lifetime; and (4) High-quality (High), which always serves the highest QoD and life time of a query.

We evaluated the performance of the scheduling schemes through simulation at this stage and take the experiments on real sensor motes as future work. In the simulation experiments, the sensory data for the queries were synthetic data, since there were no available sensory dataset for up to 100 nodes. Hence, we used the source dataset from Intel lab [11] and expanded it to up to 100 nodes using a data generation tool [20]. We fixed the WSN to be one with randomly deployed 100 nodes (including the sink) in an 100 meter * 100 meter area, in which there was at least one route from each node to the sink node. The transmission range is 25 meters (a MICA series sensor mote can reach this distance when the transmission current is about 22 mA). WSNs with such a configuration are widely used in the sensor networking studies [4], [9], [14], [24], [25].

The queries tested were acquisition and aggregation queries. To focus on the study of system profits and avoid the interference with query selectivity, in the experiments we fixed the queries as Q1 and Q2 (Section 2.

6.2. Experiment Results

In the experiments, we found the following major factors affecting the system profits from processing queries in a WSN: (1) scheduling protocol type, (2) query selectivity, (3)quality functions, (4) available resources, (5) query execution order; and (6) query type. Hence, in our experiments, we present the experiment results by varying one factor but fixing the others to get a thorough evaluation of QAS.

Scheduling Protocol. We compared QAS on the four scheduling protocols for WSNs using Q1. The query predicate is designed to make the selectivity to be about 70%. A query with a selectivity lower than 100% is the common case in both data collection and monitoring applications. Here we choose a little higher selectivity (70%) to make the protocols work in a relatively high communication traffic network to investigate their performance under a heavy workload. The quality function is as follows: $k1 = b1 = 0; k2 = -10, b2 = 100; k3 = 333.33, b3 = -133.33$. Such a quality function specifies that, for a network to attain profits, the minimum quality it should serve is as follows: the shortest query lifetime is 1 epoch, longest response time is 10s and the lowest data quality is 0.4. Such requirements are common in real world applications.

The results of the four schemes are shown in Fig. 5. As shown in this figure, different underlying scheduling protocols achieve different system profits with a given quality-aware scheme. Some scheduling protocols combined with a quality-aware scheme may get a negative profit, e.g., FPS with Random. No matter what underlying scheduling protocols used, QAS outperformed the other three quality-aware schemes. The High scheme was close to QAS on AHS, due to the highest profit was achieved near the high end of the range of the qualities. Overall, AHS achieves the highest system profits across all schemes. In the following experiments, we use AHS as the underlying scheduling protocol.

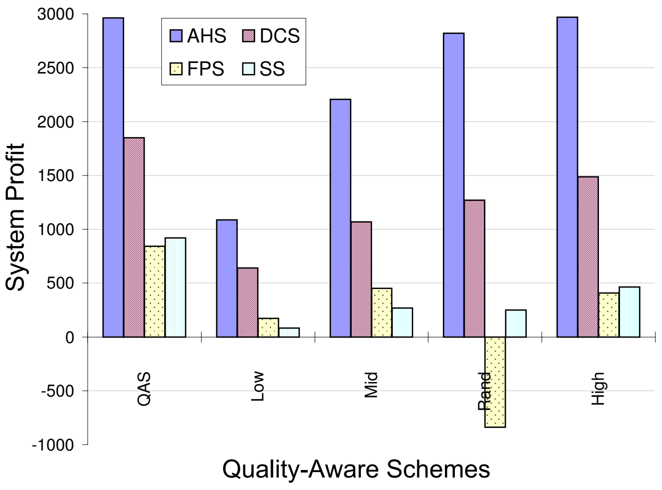


Figure 5. Different scheduling protocols

6.2.1. Selectivity. To investigate the effects of selectivities on AHS, we simulated three levels of selectivities, 0.1, 0.5 , and 1, on AHS. The quality function is $k1 = b1 = 0; k2 = -10, b2 = 100; k3 = 167.67, b3 = -66.67$. Figure 6 presents their attained system profits. In comparison, QAS on AHS almost attained the highest system profit when the query selectivity ranging from low to high. The point at which the WSN can attain the highest system profit is the lowest acceptable quality of data. Hence, the ”low” strategy attained the highest system profit at this time. Since QAS needed more overhead the determination the optimal target quality with cost modeling, the system profit attained by QAS is a little low than the highest one. Especially when the cost of query was low and then the overhead became relatively high, which made QAS perform worth, as can be seen when the selectivity is 0.1.

Due to this difference of performance on the four scheduling protocols, and on AHS with various selecitivities, In the next experiments, we set the selectivity of queries is 1 by changing the predicates in the queries. We chose DCS as the basis scheduling protocol, so that our study would focus on the remaining factors that affecting the system profits.

Quality Function. A network running the same scheduling scheme may get different system profits from a query given different quality functions. Since QAS allows a user to specify the quality function of each query, the effect of quality function on system profits should be studied.

Fig. 5 only shows the system profit from executing queries with *standard* quality functions. A standard quality function for a query refers to a function in which the revenue is

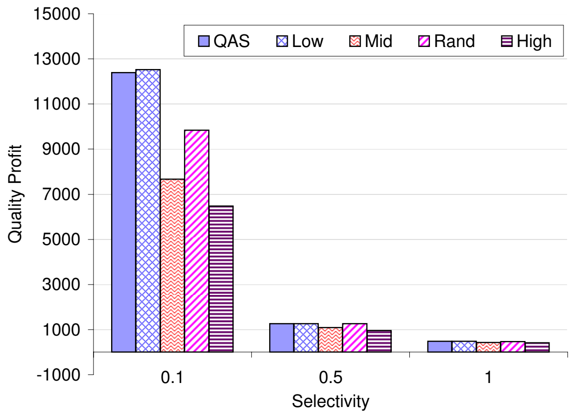
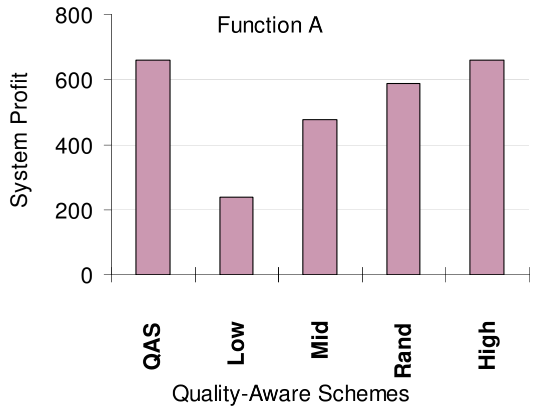


Figure 6. System Profits of queries with different selectivities



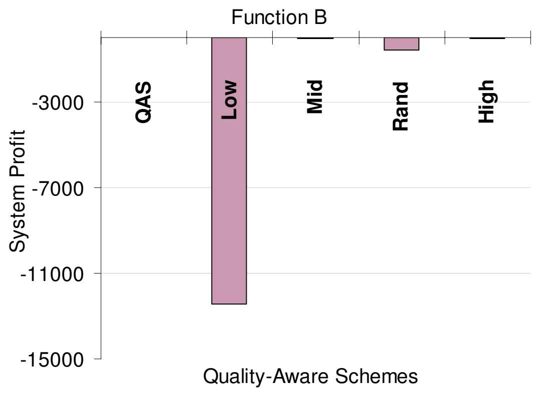


Figure 7. Q1 with different quality functions

proportional to the cost of the query. We also tested Q1 with two non-standard quality functions to evaluate QAS as shown in Table 4. These two non-standard quality functions are used to favor certain performance metrics. For instance, Function A in this table desires a high QoD and Function B prefers longer lifetime (at least 1000 epochs).

Table 4. Non-standard Quality Functions for Q1

Function	k1	b1	k2	b2	k3	b3
A	0	0	-1.	100	1000	-800
B	1	-1000	-60	600	166.7	-66.7

The attained system profits are shown in Figure 7. The results show that, QAS achieved the highest profit for both non-standard quality functions. Especially for Function B, QAS was able to avoid negative profit.

Query Execution Order. We now show the comparison of three strategies, the algorithm in QAS, a greedy algorithm, and an exhaustive searching algorithm on scheduling of multiple queries. The greedy algorithm always arranges the queries in descending order of system profits. The exhaustive algorithm traverses all of the permutations of queries and arranges the queries in the order that will get the highest total system profits.

We tested a group of five queries, where the required query life time and QoD of each were 1 and 0.4, respectively. The acceptable response time of the queries were 30s ($Q1_3$), 40s ($Q1_4$), 10s ($Q1_1$), 50s ($Q1_5$), 20s ($Q1_2$) (The query injection order was: $Q1_3$, $Q1_4$, $Q1_1$, $Q1_5$, $Q1_2$). We used different quality functions of the queries, so that different execution orders of queries would not get the same system profits.

As shown in Figure 8(a), when the quality functions specified the revenues in a way that made the order of possible highest system profit (PHSP) as $Q1_1 > Q1_2 > Q1_3 > Q1_4 > Q1_5$, then the three strategies got the same total system profits. The PHSP is the highest system profit that can be attained from processing a query given a WSN. We found that the strategies determined the same query execution order and it is the optimal query execution order.

However, when the quality functions did not enable the PHSP of $Q1_1$ to be the largest one, the execution orders given by the strategies were much different. As demonstrated in Figure 8, in general the performance of QAS was better than the greedy algorithm and close to the exhaustive algorithm. In the worst case, QAS attained the same system profit as that of the greedy algorithm.

6.2.2. Available Resources. In query processing on a WSN, it is often found that the quality request of a query is not satisfied although the query results are returned, especially when there are multiple queries running in the network or the quality request is too high. The underlying reason is that the resources are not enough to satisfy the user requests and the query processors fail to utilize the available resources to satisfy the request as much as possible, which causes some resources to be wasted in serving additional qualities not

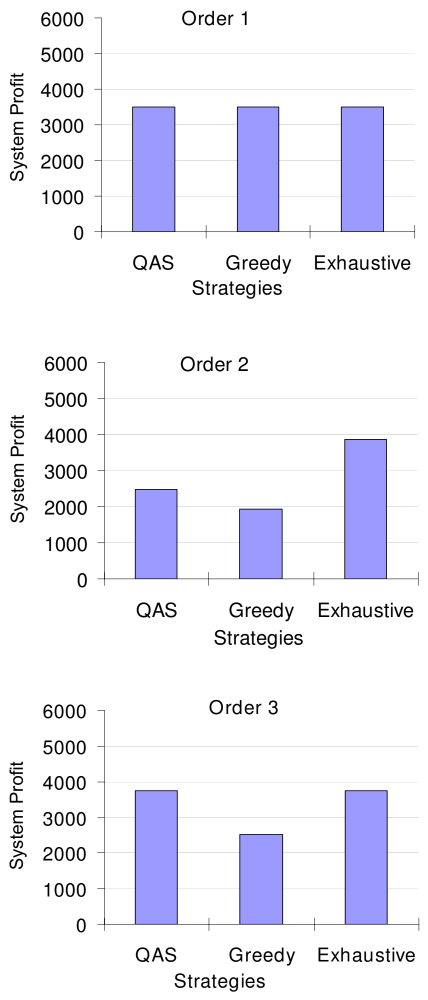


Figure 8. System Profits of the three strategies

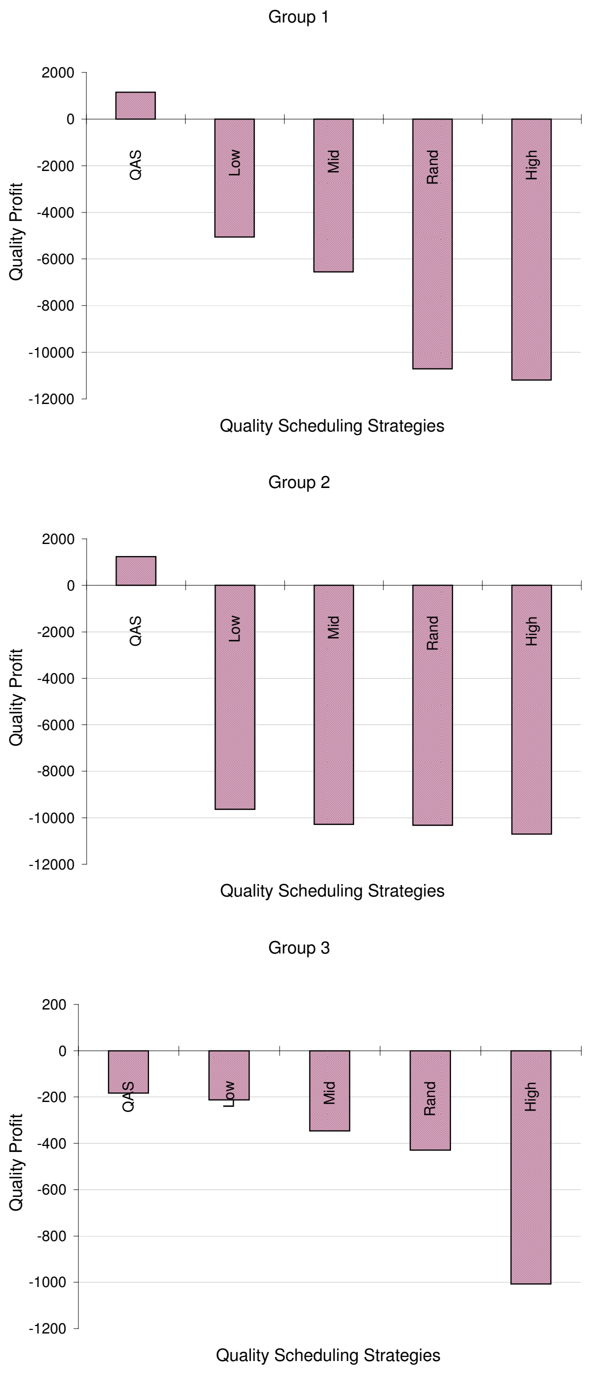


Figure 9. Profits of Groups of Queries

required by the users. Since wireless sensor networks are highly resource limited networks, this resource limitation problem is much important in quality aware scheduling. In the following, we evaluate the performance of QAS on limited resources that are not enough for all queries injected.

- Group 1: 5 queries, query life time of each is 1; the acceptable response time are 50s, 40s, 30s, 20s, and 10s, respectively; the minimal QoD is 0.4.
- Group 2: 5 queries, query life time of each is 1; the acceptable response time are 50s, 40s, 30s, 20s, and 10s, respectively; the minimal QoD is 0.8.
- Group 3: one query, query life time of each is 1; the acceptable response time is 10s, respectively; the minimal QoD is 0.8.

6.2.3. Query Type. To study quality scheduling on different types of queries, we evaluate QAS on the aggregation query,

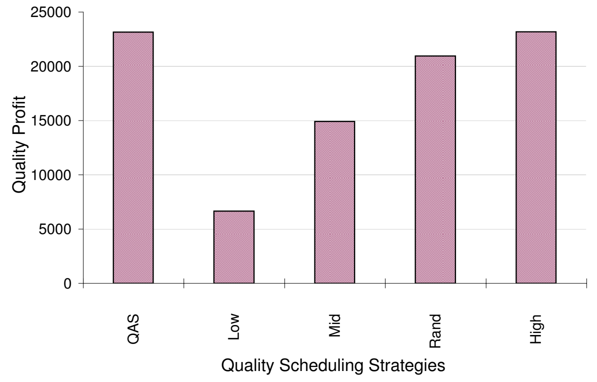


Figure 10. Aggregation query

Q2, in this subsection. The quality function of Q2 is shown in Table 5. The highest revenue to Q2 is 100, the same as that of Q1 in Figure 5.

Table 5. Quality Function of Q2

Quality Function	**k1**	**b1**	**k2**	**b2**	**k3**	**b3**
f-Q2	0	0	**-10**	**100**	166.7	-66.7

The system profits of QAS and other strategies from running Q2 are shown in Figure 10. The result show that although the Q2 was given the same level of revenue as that of Q1 in Figure 6, the aggregation query could get much higher system profit (22800 vs. 3000). The reason is that in processing an aggregation query, the internal nodes only sent one data packet. In contrast, they needed to send multiple data packets to forward the query results of their descendants in an acquisition query. Moreover, the results show that QAS still attained the maximum system profit in process aggregation queries due to its quality aware scheduling.

6.2.4. Profit within Network Lifetime. Finally, we evaluated the system profits of a WSN in its network lifetime. In this paper, we define the network lifetime as the time from a network starts to the time that the first node runs out of energy. We measured the total profits of various queries that are injected and removed in the WSN during its lifetime. The queries and their quality functions were randomly chosen from a query pool, in which the quality requirement on each query was with the range of $L \in [1, 10000]$,$D \in [0.1, 1]$, $T \in [0, 60000]$, where L, D, T are query lifetime, data quality, and response time, respectively.

Figure 11 shows the detailed accumulated system profits of QAS on AHS. The figure shows that, within the lifetime of a network, the scheduling protocol with QAS can attain more profits than others. QAS even outperforms the "Low"

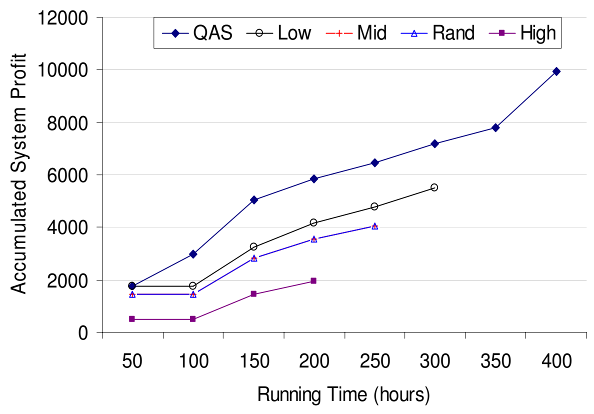


Figure 11. Accumulated system profits over time

strategy in terms of lifetime, which keeps serving the lowest possible quality during the network lifetime to save the cost. The reason is that the "Low" strategy cannot get profits when the network fail to serve the lowest quality requirements from the strict-requirement queries. In contrast, QAS will not serve these queries if it finds the resources are not enough. This way, QAS avoids the energy wastes.

7. Conclusion and Future Work

In this paper, we presented a quality aware scheduling framework, QAS, which efficiently satisfies the user quality requirements on queries. QAS runs on existing quality-unaware scheduling protocols of WSNs and enables users to specify their quality requirements using quality functions. Given these quality functions, QAS determines the target quality of each query, at which the ratio between the revenue and the energy cost is maximal.

QAS effectively solves the energy waste problem in sensor query processing that causes high quality requirement queries to be unsatisfied but low quality requirement queries to be overly satisfied. As shown in the experimental results, QAS outperforms the baseline quality scheduling strategies.

Acknowledgment

This work was supported by grant 617307 from the Hong Kong Research Grants Council and by NSF Career award #IIS 0746696.

References

[1] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Çetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. B. Zdonik, "The design of the borealis stream processing engine," in *CIDR*, 2005.

[2] R. K. Abbott and H. Garcia-Molina, "Scheduling real-time transactions: A performance evaluation," *ACM Trans. Database Syst.*, vol. 17, no. 3, pp. 513–560, 1992.

[3] M. Amirijoo, S. Son, and J. Hansson, "Qod adaptation for achieving lifetime predictability of wsn nodes communicating over satellite links," in *INSS*, June 2007.

[4] Q. Cao, T. F. Abdelzaher, T. He, and J. A. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare-event detection," in *IPSN*, 2005.

[5] H.-R. Chen and Y.-H. Chin, "An adaptive scheduler for distributed real-time database systems," *Inf. Sci.*, vol. 153, pp. 55–83, 2003.

[6] Crossbow, "http://www.xbow.com."

[7] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *SenSys*, I. F. Akyildiz, D. Estrin, D. E. Culler, and M. B. Srivastava, Eds. ACM, 2003, pp. 138–149.

[8] J. R. Haritsa, M. J. Carey, and M. Livny, "Value-based scheduling in real-time database systems," *VLDB J.*, vol. 2, no. 2, pp. 117–152, 1993.

[9] B. Hohlt, L. Doherty, and E. A. Brewer, "Flexible power scheduling for sensor networks," in *IPSN*, 2004.

[10] D. Hong, T. Johnson, and S. Chakravarthy, "Real-time transaction scheduling: A cost conscious approach," in *SIGMOD*, 1993.

[11] IntelLabData, "http://berkeley.intel-research.net/labdata."

[12] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel, "Delay efficient sleep scheduling in wireless sensor networks," in *INFOCOM*. IEEE, 2005.

[13] S. Madden., M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks," in *OSDI*, 2002.

[14] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tinydb: an acquisitional query processing system for sensor networks," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 122–173, 2005.

[15] H. Pang, M. J. Carey, and M. Livny, "Multiclass query scheduling in real-time database systems," *IEEE Trans. Knowl. Data Eng.*, vol. 7, no. 4, pp. 533–551, 1995.

[16] L. Peng and K. S. Candan, "Data-quality guided load shedding for expensive in-network data processing," in *ICDE*, 2007.

[17] H. Qu and A. Labrinidis, "Preference-aware query and update scheduling in web-databases," in *ICDE*, 2007, pp. 356–365.

[18] Q. Ren and Q. Liang, "Energy and quality aware query processing in wireless sensor database systems," *Inf. Sci.*, vol. 177, no. 10, pp. 2188–2205, 2007.

[19] K. Römer and F. Mattern, "The design space of wireless sensor networks," *IEEE Wireless Communications*, vol. 11, no. 16, pp. 54–61, 2004.

[20] SDGEN, "http://www.cse.ust.hk/catalac/."

[21] M. L. Sichitiu, "Cross-layer scheduling for power efficiency in wireless sensor networks," in *INFOCOM*, 2004.

[22] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: a survey," *IEEE Network*, vol. 18, no. 4, pp. 45–50, 2004.

[23] H. Wu and Q. Luo, "Adaptive holistic scheduling for query processing in sensor networks," *HKUST-CSE Technical Report, http://www.cse.ust.hk/tahoe*, April, 2007.

[24] H. Wu, Q. Luo, and W. Xue, "Distributed cross-layer scheduling for in-network sensor query processing," in *PerCom*. IEEE Computer Society, 2006, pp. 180–189.

[25] Y. Yao and J. Gehrke, "Query processing in sensor networks," in *CIDR*, 2003.

[26] D. Yates, E. Nahum, J. Kurose, and P. Shenoy, "Data quality and query cost in wireless sensor networks," in *Pervasive Computing and Communications Workshops*, March 2007.

[27] W. Ye, J. S. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *INFOCOM*, 2002.